Chapter 11

Message Integrity and Message Authentication

Objectives

- To define message integrity
- To define message authentication
- To define criteria for a cryptographic hash function
- To define the Random Oracle Model and its role in evaluating the security of cryptographic hash functions
- To distinguish between an MDC and a MAC
- To discuss some common MACs

11-1 MESSAGE INTEGRITY

The cryptography systems that we have studied so far provide secrecy, or confidentiality, but not integrity. However, there are occasions where we may not even need secrecy but instead must have integrity.

Topics discussed in this section:

11.1 Document and Fingerprint
11.2 Message and Message Digest
11.3 Difference
11.4 Checking Integrity
11.5 Cryptographic Hash Function Criteria

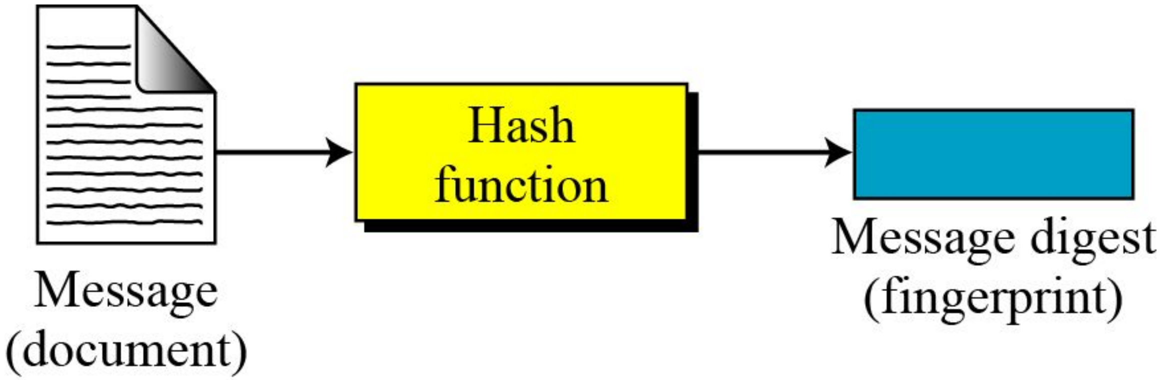
11.1.1 Document and Fingerprint

One way to preserve the integrity of a document is through the use of a fingerprint. If Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document.

11.1.2 Message and Message Digest

The electronic equivalent of the document and fingerprint pair is the message and digest pair.

Figure 11.1 *Message and digest*



11.1.3 Difference

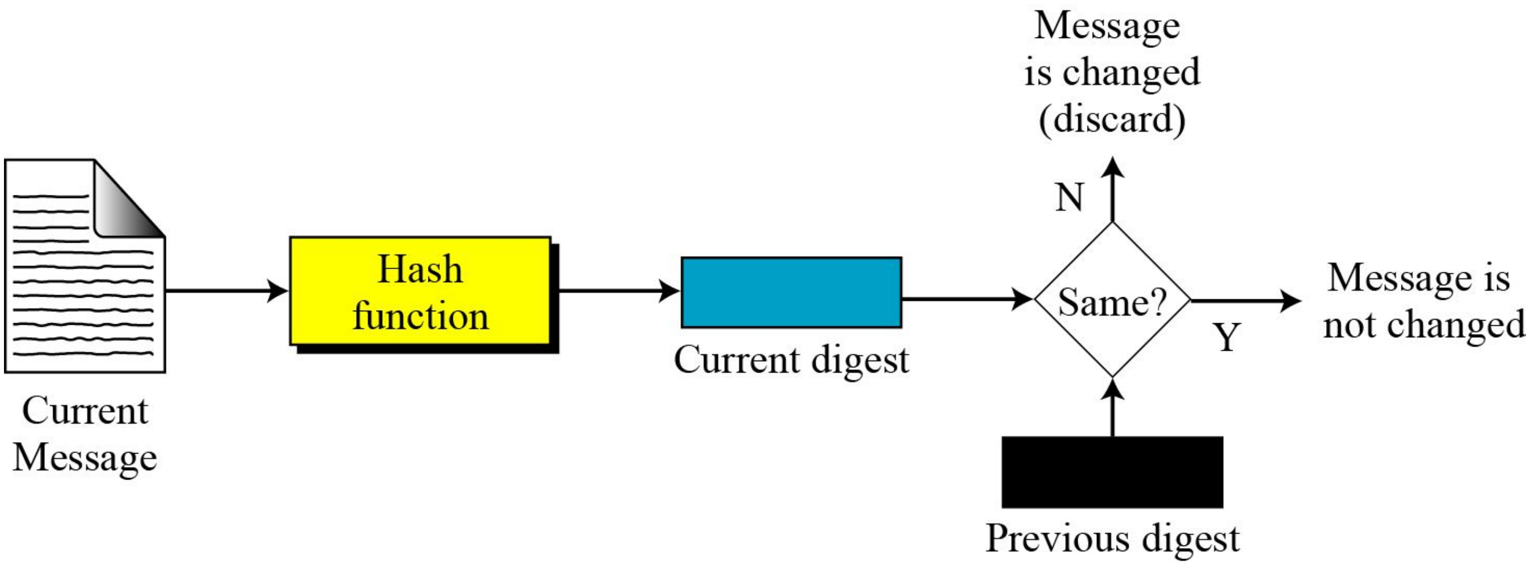
The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences. The document and fingerprint are physically linked together. The message and message digest can be unlinked separately, and, most importantly, the message digest needs to be safe from change.

Note

The message digest needs to be safe from change.

11.1.4 Checking Integrity

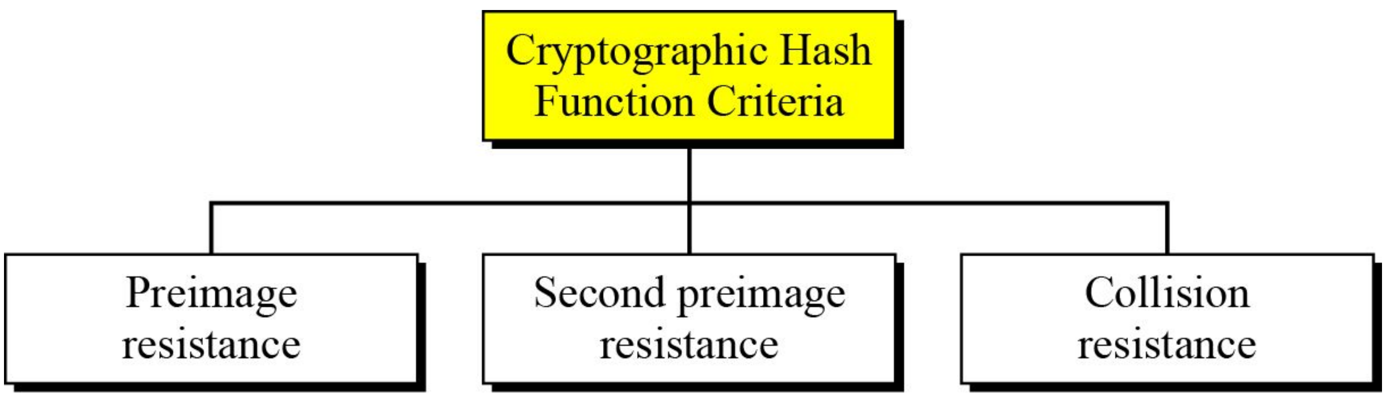
Figure 11.2 ***Checking integrity***



11.1.5 Cryptographic Hash Function Criteria

A cryptographic hash function must satisfy three criteria: preimage resistance, second preimage resistance, and collision resistance.

Figure 11.3 *Criteria of a cryptographic hash function*



Preimage Resistance

Preimage Attack

Given: y = h(M) **Find: M′ such that y = h(M′)**

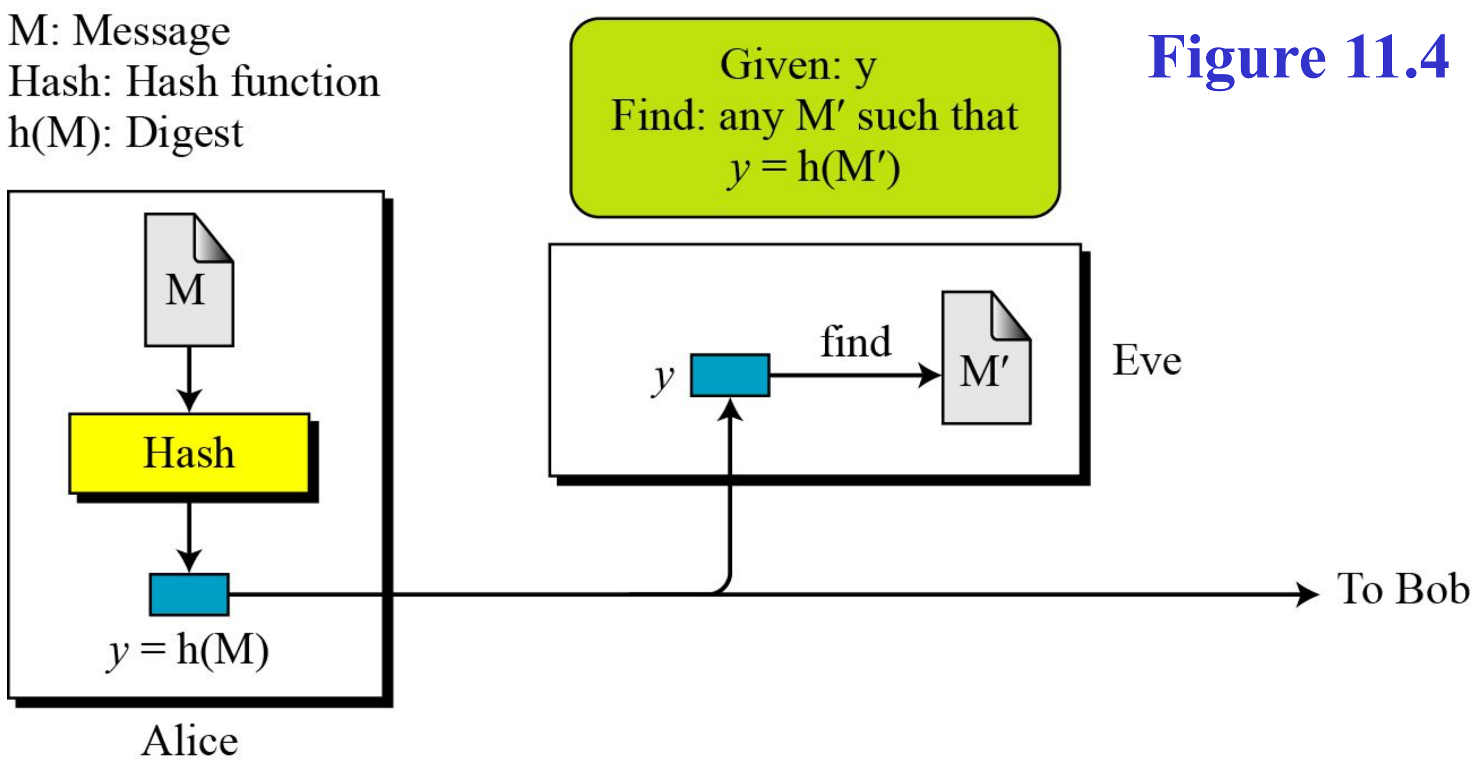


Figure 11.4 ***Preimage***

Example 11.1

Can we use a conventional lossless compression method such as *StuffIt* as a cryptographic hash function?

Solution

We cannot. A lossless compression method creates a compressed message that is reversible.

Example 11.2

Can we use a checksum function as a cryptographic hash function?

Solution

We cannot. A checksum function is not preimage resistant, Eve may find several messages whose checksum matches the given one.

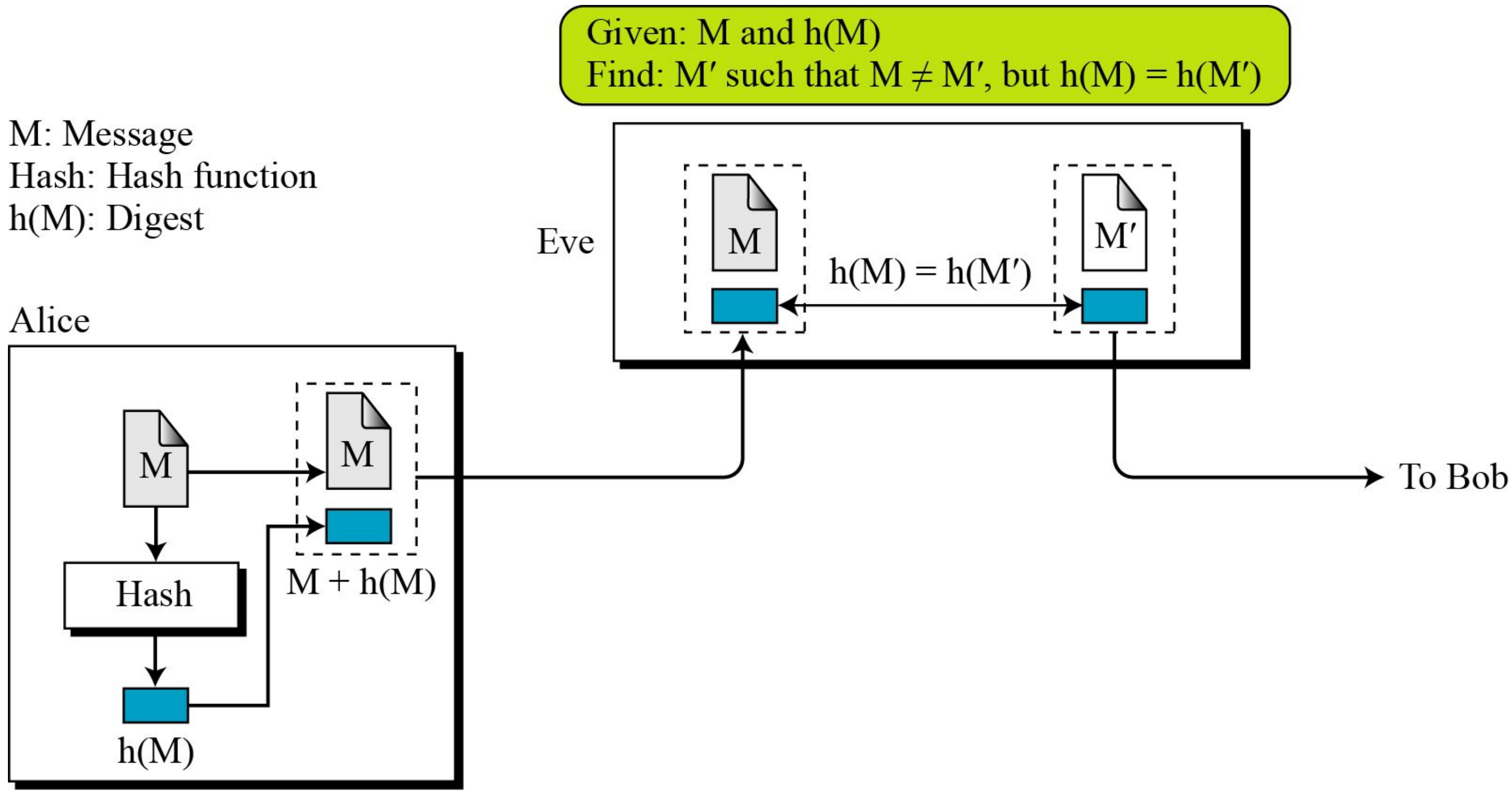
11.1.5 Continued

Second Preimage Resistance

Second Preimage Attack

Given: M and h(M) **Find: M′ ≠ M such that h(M) = h(M′)**

Figure 11.5 ***Second preimage***

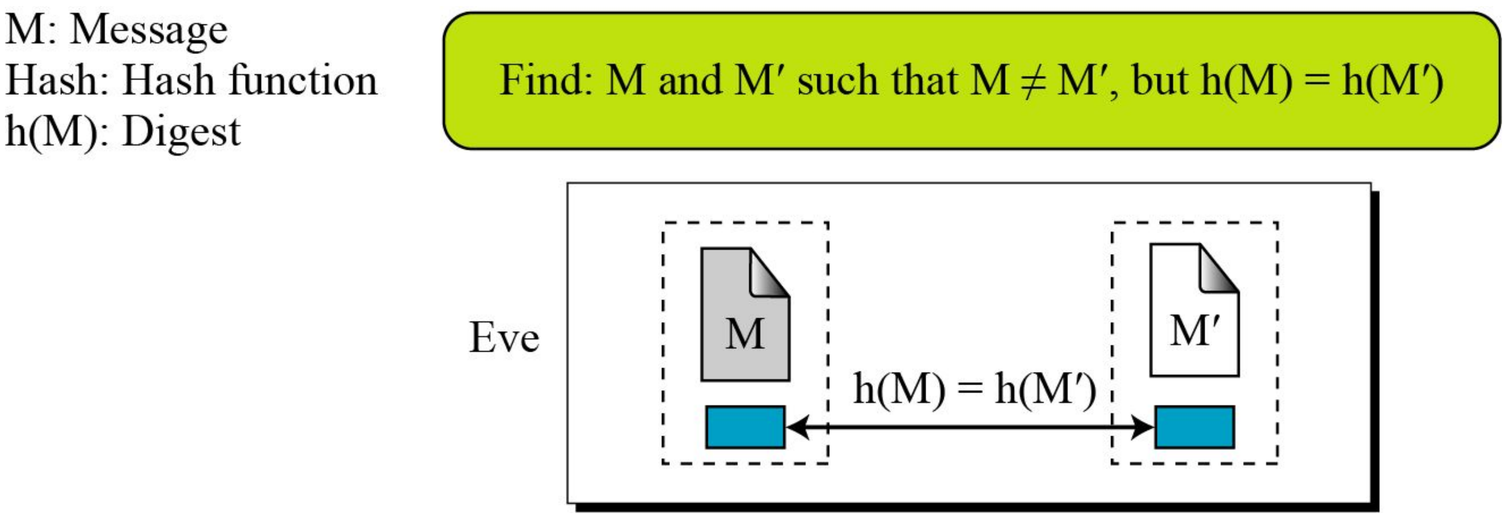


Collision Resistance

Collision Attack

Given: none **Find: $M' \neq M$ such that $h(M) = h(M')$**

Figure 11.6 ***Collision***



11-2 RANDOM ORACLE MODEL

The Random Oracle Model, which was introduced in 1993 by Bellare and Rogaway, is an ideal mathematical model for a hash function.

Topics discussed in this section:

11.2.1 Pigeonhole Principle
11.2.2 Birthday Problems
11.2.3 Attacks on Random Oracle Model
11.2.4 Attacks on the Structure

11-2 Continued

Example 11.3

Assume an oracle with a table and a fair coin. The table has two columns.

Table 11.1 *Oracle table after issuing the first three digests*

Message	*Message Digest*
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

a. The message AB1234CD8765BDAD is given for digest calculation. The oracle checks its table.

11-2 Continued

Example 11.3 Continued

Table 11.2 *Oracle table after issuing the fourth digest*

Message	*Message Digest*
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB1234CD8765BDAD	DCB1
AB45CD1048765412AAAB6662BE	A38B

b. The message 4523AB1352CDEF45126 is given for digest calculation. The oracle checks its table and finds that there is a digest for this message in the table (first row). The oracle simply gives the corresponding digest (13AB).

11-2 Continued

Example 11.4

The oracle in Example 11.3 cannot use a formula or algorithm to create the digest for a message. For example, imagine the oracle uses the formula h(M) = M mod n. Now suppose that the oracle has already given h(M1) and h(M2). If a new message is presented as M3 = M1 + M2, the oracle does not have to calculate the h(M3). The new digest is just [h(M1) + h(M2)] mod n since

$$h(M_3) = (M_1 + M_2) \bmod n = M_1 \bmod n + M_2 \bmod n = [h(M_1) + h(M_2)] \bmod n$$

This violates the third requirement that each digest must be randomly chosen based on the message given to the oracle.

11.2.1 Pigeonhole Principle

If n pigeonholes are occupied by n + 1 pigeons, then at least one pigeonhole is occupied by two pigeons. The generalized version of the pigeonhole principle is that if n pigeonholes are occupied by kn + 1 pigeons, then at least one pigeonhole is occupied by k + 1 pigeons.

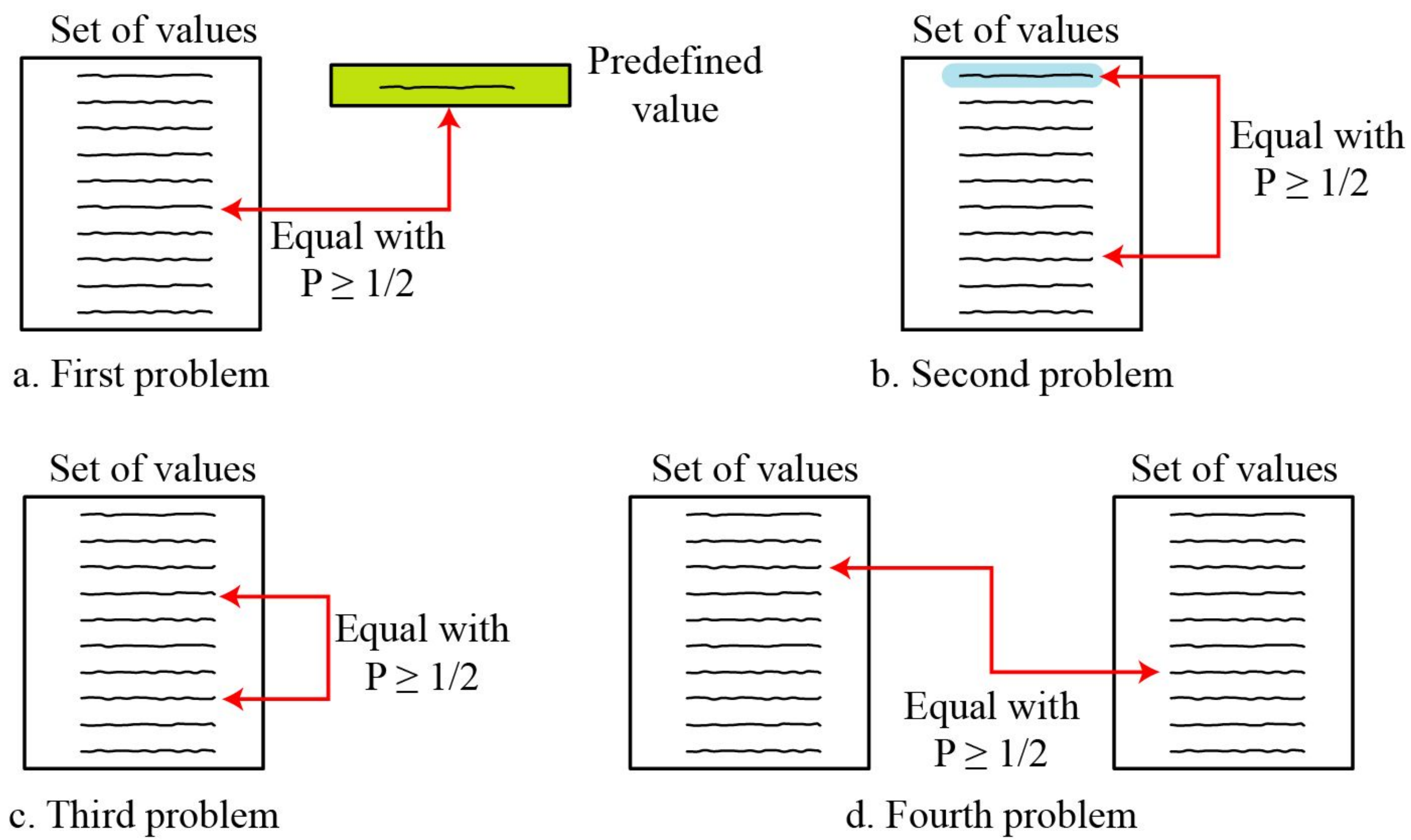
11.2.1 Continued

Example 11.5

Assume that the messages in a hash function are 6 bits long and the digests are only 4 bits long. Then the possible number of digests (pigeonholes) is 24 = 16, and the possible number of messages (pigeons) is 26 = 64. This means *n* = 16 and *kn* + 1 = 64, so *k* is larger than 3. The conclusion is that at least one digest corresponds to four (*k* + 1) messages.

11.2.2 Birthday Problems

Figure 11.7 ***Four birthday problems***



Example 11.9

MD5 (see Chapter 12), which was one of the standard hash functions for a long time, creates digests of 128 bits. To launch a collision attack, the adversary needs to test 2^{64} ($2^{128}/2$) tests in the collision algorithm. Even if the adversary can perform 2^{30} (more than one billion) tests in a second, it takes 2^{34} seconds (more than 500 years) to launch an attack. This type of attack is based on the Random Oracle Model. It has been proved that MD5 can be attacked on less than 2^{64} tests because of the structure of the algorithm.

11.2.3 Continued

Example 11.10

SHA-1 (see Chapter 12), a standard hash function developed by NIST, creates digests of 160 bits. The function is attacks. To launch a collision attack, the adversary needs to test $2^{160/2} = 2^{80}$ tests in the collision algorithm. Even if the adversary can perform 2^{30} (more than one billion) tests in a second, it takes 2^{50} seconds (more than ten thousand years) to launch an attack. However, researchers have discovered some features of the function that allow it to be attacked in less time than calculated above.

Example 11.11

The new hash function, that is likely to become NIST standard, is SHA-512 (see Chapter 12), which has a 512-bit digest. This function is definitely resistant to collision attacks based on the Random Oracle Model. It needs $2^{512/2} = 2^{256}$ tests to find a collision with the probability of 1/2.

11.2.4 Attacks on the Structure

The adversary may have other tools to attack hash function. One of these tools, for example, is the meet-in-the-middle attack that we discussed in Chapter 6 for double DES.

11-3 MESSAGE AUTHENTICATION

A message digest does not authenticate the sender of the message. To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not an impostor. The digest created by a cryptographic hash function is normally called a modification detection code (MDC). What we need for message authentication is a message authentication code (MAC).

Topics discussed in this section:

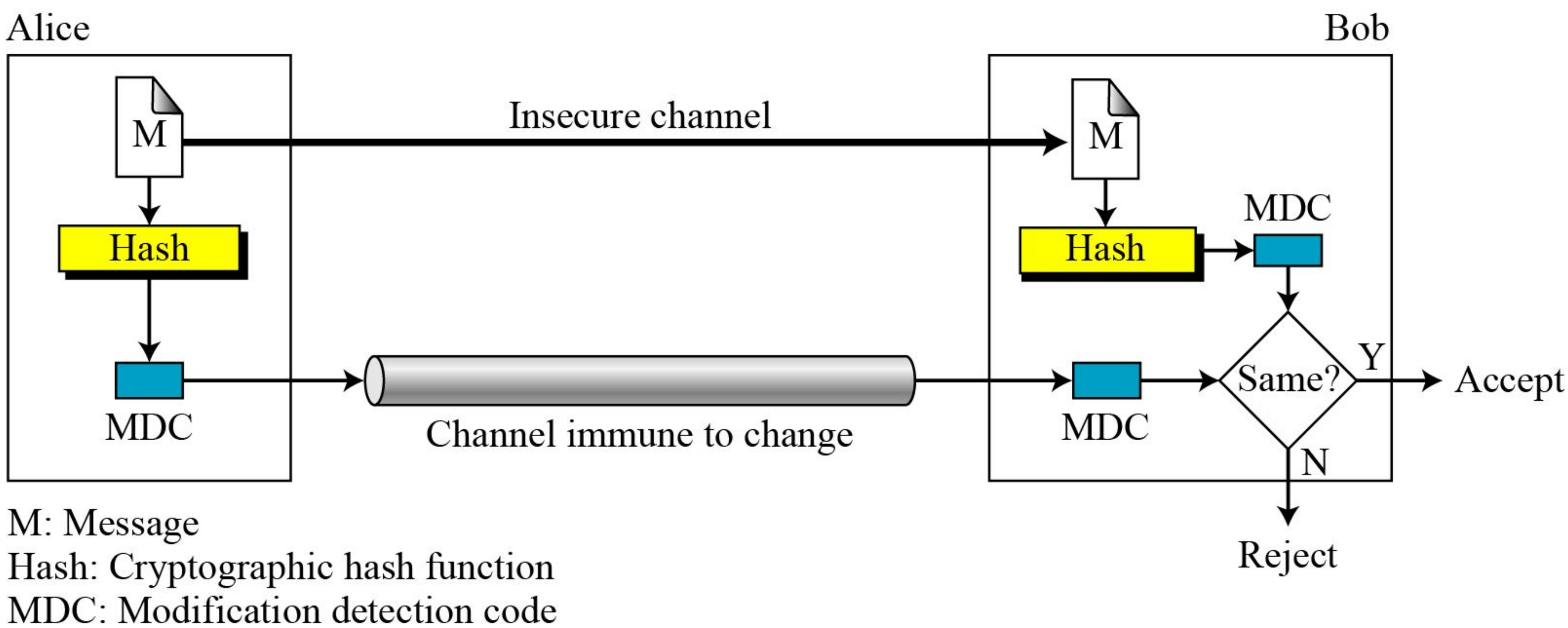
11.3.1 Modification Detection Code (MDC)
11.3.2 Message Authentication Code (MAC)

11.3.1 Modification Detection Code (MDC)

A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed. If Alice needs to send a message to Bob and be sure that the message will not change during transmission, Alice can create a message digest, MDC, and send both the message and the MDC to Bob. Bob can create a new MDC from the message and compare the received MDC and the new MDC. If they are the same, the message has not been changed.

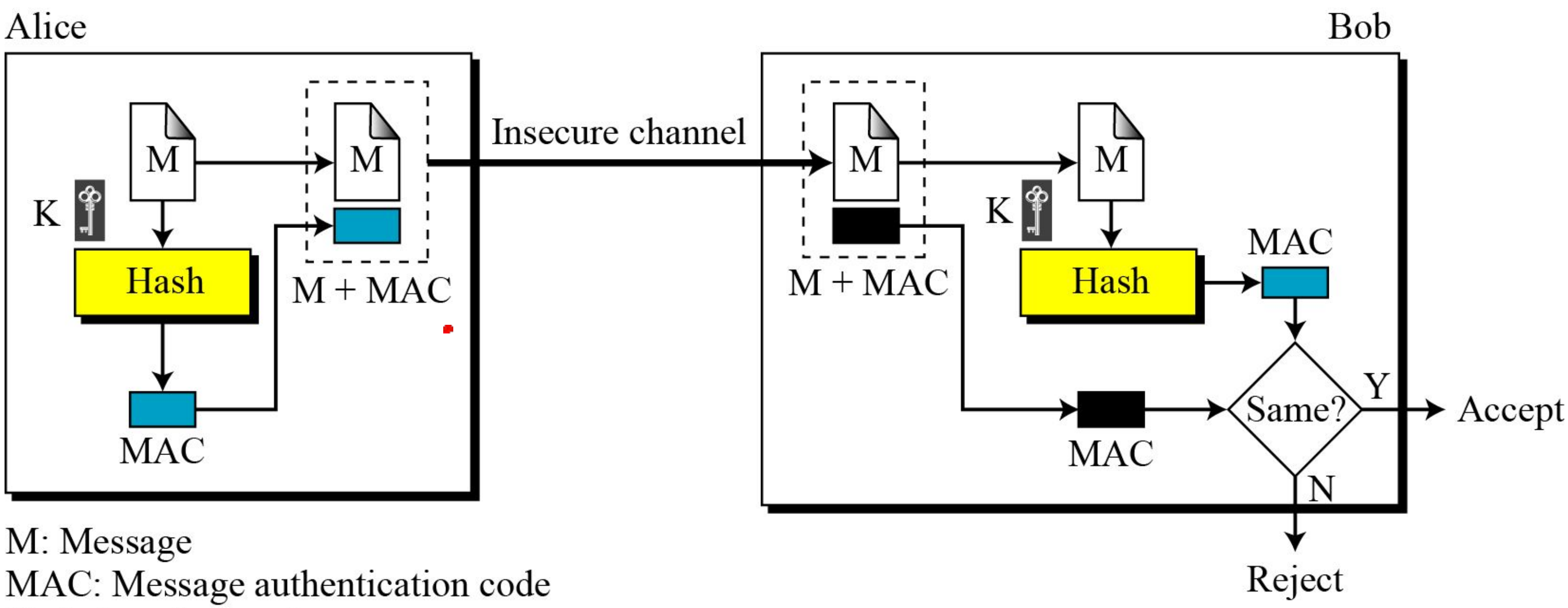
11.3.1 Continued

Figure 11.9 ***Modification detection code (MDC)***



11.3.2 Message Authentication Code (MAC)

Figure 11.10 ***Message authentication code***



M: Message
MAC: Message authentication code
K: A shared secret key

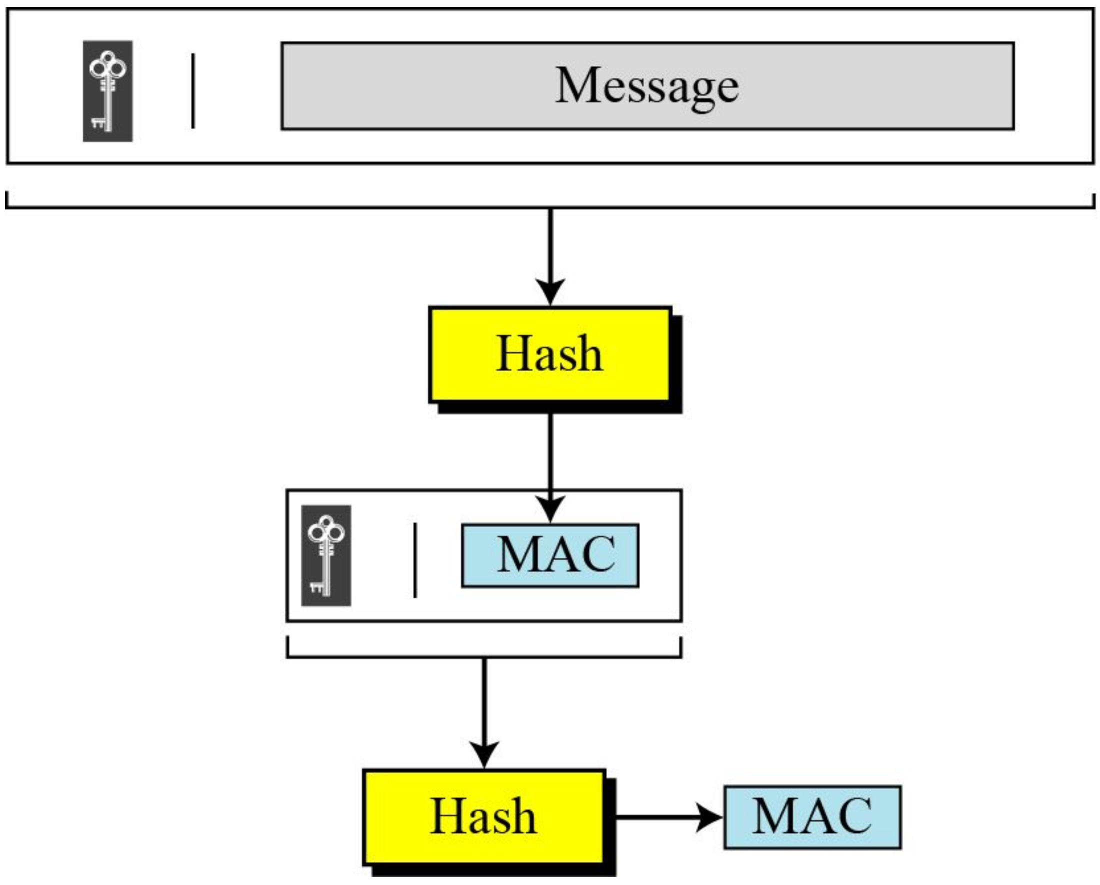
11.3.2 Continued

Note

The security of a MAC depends on the security of the underlying hash algorithm.

Nested MAC

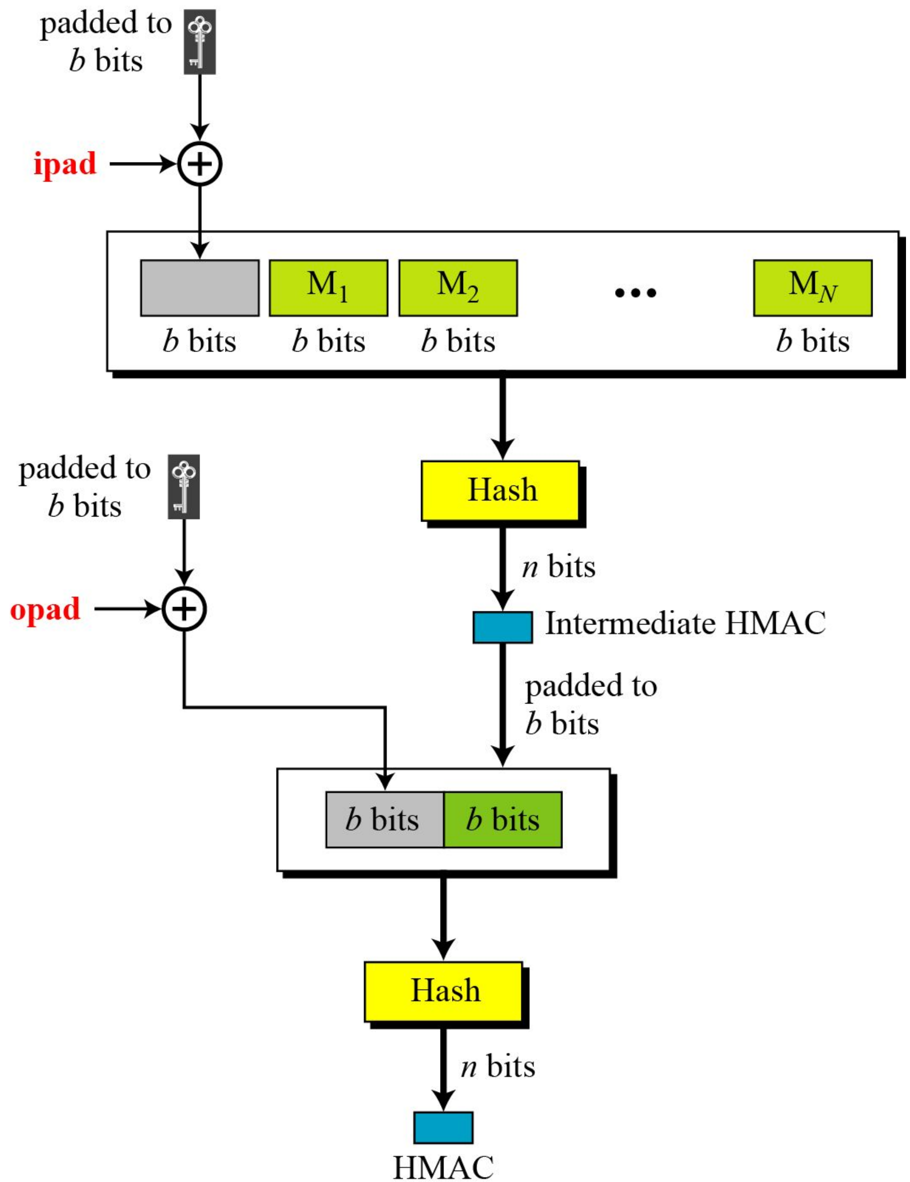
Figure 11.11 *Nested MAC*



11.3.2 Continued

HMAC

Figure 11.12
Details of HMAC



11.3.2 Continued

Figure 11.13 *CMAC*